3 9

MEASURING THE EFFECTS OF HETEROGENEITY ON DISTRIBUTED SYSTEMS

Mohamed El-Toweissy Osman ZeinElDine Ravi Mukkamala

Department of Computer Science
Old Dominion University
Norfolk, Virginia 23529

ABSTRACT

Distributed computer systems in daily use are becoming more and more heterogeneous. Currently, much of the design and analysis studies of such systems assume homogeneity. This assumption of homogeneity has been mainly driven by the resulting simplicity in modeling and analysis. In this paper, we present a simulation study to investigate the effects of heterogeneity on scheduling algorithms for hard real-time distributed systems. In contrast to pervious results which indicate that random scheduling may be as good as a more complex scheduler, our algorithm is shown to be consistently better than a random scheduler. This conclusion is more prevalent at high workloads as well as at high levels of heterogeneity.

INTRODUCTION

With the advancing communication technologies and the need for integration of global systems, heterogeneity is becoming a reality in distributed computer systems. However, most existing performance studies of such systems still assume homogeneity; be it in hardware (e.g., node speed) or in software (e.g., scheduling algorithms). Generally, such homogeneity assumptions are dictated by the resulting simplicity in modeling and analysis.

Clearly, heterogeneous systems are less analytically tractable than their homogeneous counterparts. Typically, heterogeneity will result in increased number of variables in the context of analytical techniques such as mathematical programming, probabilistic analysis, and queuing theory. This is one reason for assuming homogeneity while using analytical techniques. In the case of simulation techniques, however, it is possible for a modeler to introduce any level of heterogeneity into the system. The problem now lies in the complexity of interpretation of the results. If the simulator was written with the basic objective of testing a hypothesis or comparing the performance of a set of algorithms, introducing heterogeneity will substantially increase efforts to separate its effects from those of the algorithm. Thus, a modeler is more likely to assume a homogeneous system.

With this in mind, we have been investigating into the effects of heterogeneity on the performance of distributed systems. Our initial efforts, reported in (ZeinElDine et al. 1991) and this paper, focus on scheduling in hard real-time systems. For this purpose, we have designed a distributed scheduler aimed at handling various heterogeneities; in particular, heterogeneities in nodes, node traffic and local scheduling algorithms.

In the rest of this paper we present the system model. Next, we discuss some issues related to the effectiveness of our algorithm. Major results with their respective conclusions are then portrayed. Finally, we highlight some recommendations for future work.

THE PROPOSED MODEL

For the purposes of scheduling, the distributed system is modeled as a tree of nodes as is shown in Figure 1. The nodes at the lowest level (level 0) are the *processing* nodes while the nodes at the higher levels represent *servers* (or guardians). A processing node is responsible for executing arriving jobs when they meet some specified criteria (e.g., deadline). The processing nodes are grouped into *clusters*, and each cluster is assigned a unique server. When a server receives a job, it tries to either redirect that job to a processing node within its cluster or to its guardian. It is to be noted that this hierarchical structure could be logical (i.e., some of the processing nodes may themselves assume the role of the servers).

The system model has four components: jobs, processing nodes, servers, and the communication subsystem. A job is characterized by its arrival time, execution time, deadline, and priority (if any). The specifications of a processing node include its speed factor, scheduling policy, external arrival rate (of jobs), and job mix (due to heterogeneity). A server is modeled by its speed and its node assignment policy. Finally, the communication subsystem is represented by the speeds of transmission and distances between different nodes (processing and servers) in the system.

Operation

The flow diagram of the scheduling algorithm is shown in Figure 2. When a job with deadline arrives (either from an external user or from a server) at a processing node, the local scheduling algorithm at the node decides whether or not to execute this job locally. This decision is based on pending jobs in the local queue (which are already guaranteed to be executed within their deadlines), the requirements of the new job, and the scheduling policies (e.g., FCFS, SJF, SDF, SSF etc. (Zhao et al. 1987)). In case the local scheduler cannot execute the new job, it either sends the job to its server (if there is a possibility of completion), or discard the job (if there is no such chance of completion).

The level-1 server maintains a copy of the latest information provided by each of its child nodes including the load at the node and its scheduling policy. Using this information, the server should be able to decide which processing nodes are eligible for executing a job and meet its deadline. When more than one candidate node is available, a random selection is carried out among these nodes. If a server cannot find a candidate node for executing the job, it forwards the job to the level-2 server.

The information at the level-2 server consists of an abstraction of the information available at each of the level-1 servers. This server redirects an arriving job to one of the level-1 servers. The choice of candidate servers is dependent on the ability of these servers to redirect a job to one of the processing nodes in their cluster to meet the deadline of the job. (For more details on operation and information contents at each level, the reader is advised to refer to (ZeinElDine et al. 1991)).

EXPERIMENT

In order to utilize the proposed scheduler as a vehicle for our research on measuring the effects of heterogeneity, first it has to be proven effective. Consequently, we have conducted several parametric studies to determine the sensitivity of our algorithm to various parameters: the cluster size, the frequency of propagation of load statistics (between levels), the processing node scheduling policy (FCFS, SJF etc), the communication delay (between nodes), and the effects of information structures. For lack of space, we present a sample of the results pertaining to the first three of these parameters. Accordingly, all the results reported here assume:

- the total number of processing nodes is 100;
- equal load at all nodes;
- communication delay between any nodes is the same.

The performance of the scheduler is measured in terms of the percentage of jobs discarded by the algorithm (at levels 0, 1 & 2). The rate of arrivals of jobs and their processing requirements are combinedly represented through a load factor. This load factor refers to the load on the overall system. Our load consists of jobs from three types of execution time constraints (10, 50 & 100) with slack (25, 35 & 300) respectively.

Discussion

We now discuss our observations regarding the characteristics of the distributed scheduling algorithm (DSA) in terms of the three selected parameters. In order to isolate the effect of one factor from others, the choice of parameters is made judiciously. For example, in studying the effects of cluster size (Figure 3), the updation period is chosen to be a medium value of 200 (stat=200). For each parametric study we have two sets of runs, they differ in the local scheduling policy at the processing nodes; one set uses FCFS while the the other uses SJF.

Cluster size Cluster size indicates the number of processing nodes being assigned to a level-1 server. In our study, we have considered three cluster sizes: 100, 50, and 10. A cluster of 100 nodes indicates a centralized server structure where all the processing nodes are under one level-1 server. In this case, level-2 server is absent. Similarly, in the case of cluster of 50 nodes, there are two level-1 servers, and one level-2 server. For 10-node cluster, we have 10 level-1 servers. In addition, we consider a completely decentralized case represented by the *random* policy. In this case, each processing node acts as its own server and randomly selects a destination node to execute a job which it cannot locally guarantee.

The results are plotted in Figure 3. These results show that our algorithm is robust to variations in

cluster size. In addition, its performance is significantly superior to a random policy.

Frequency of updations The currency of information at a node about the rest of the system plays a major role in performance. Hence, if the state of processing nodes varies rapidly, then the frequency of status information exchange between the levels should also be high. In order to determine the sensitivity of the proposed algorithm to the period of updating statistics at the servers, we experimented with four time periods: 25, 100, 200 and 500 units. The results are summarized in Figure 4. From these results, the following observations are drawn:

- our algorithm is extremely sensitive to changes in period of information exchanges between servers and processing nodes;
- even in the worst case of 500 units, the performance of our algorithm is significantly better than the random policy.

Local scheduling policy Our third parametric study is concerned with the effect of the scheduling policy at the processing nodes. In this paper, we report the results of the runs conducted with all the processing nodes having the same scheduling policy; either FCFS or SJF. Later in our work, we plan to experiment with different mixes of local scheduling policies. We would also give each node the freedom to select its scheduling policy in order to determine the impact of node autonomy on the overall performance of the system. This issue is of crucial importance, since there is no scheduling policy that best fits all working environments.

Revisiting the results of Figures 3 and 4, we can observe the following:

- both FCFS and SJF behave similarly at light to moderate loads, while SJF is consistently better than FCFS at high loads;
- the percentage of discarded jobs sharply increases with the increase in the load factor for both the Random and FCFS policies. However, for SJF the rate of increase in the percentage of discarded jobs dramatically drops at high loads.

The reason for the above result is that, at light to moderate loads there is no build up at the processing node queues, consequently, the dominant factor is the jobs being processed at the node processors. This behavior is the same for all policies. However, when the queues start to build up, the respective queue policy prevails. Hence, for the SJF, the short jobs with their relatively small slack, will have a better probability of being executed.

EFFECTS OF HETEROGENEITY

So far, our concentration has been on gaining better insight into the behavior of our algorithm in order to assess its viability and suitability to be able to conduct further research. Proven effective, we return back to the objective for which the algorithm has been developed. The main goal of the current phase of our studies is measuring the effects of heterogeneity on scheduling in hard real-time distributed systems. For this purpose, we are pursuing multiple experiments to measure the effects of node heterogeneity (simply represented by node speed), heterogeneity in scheduling algorithms. heterogeneity in loads as well as other system heterogeneities. In this paper, we present the effects of node heterogeneity. (Results on other types will be reported in a sequel of papers).

We consider four different node speed distributions (het1, het2, het3, and hom). The homogeneous case (denoted by hom) represents a system with 100 nodes having the same unit speeds. The three heterogeneous case are represented by het1, het2, and het3. Each of these set are described by a set of <# of nodes, speed factors> pairs. The average speed factor for all distribution is 1.0, so the average system speed is the same. The three heterogeneous case differ in their speed factor variance, thus varying the degree of heterogeneity. While het3 represents a severe case of heterogeneity, het1 is more biased towards homogeneity.

The results are included in Figure 5. From these results, we observe that:-

- with our algorithm, even though the increase in degree of heterogeneity resulted in an increase of discarded jobs, the increase is not so significant. Hence, our algorithm appears to be robust to node heterogeneities.
- the performance of the random policy is extremely sensitive to the node heterogeneity. As the heterogeneity is increased, the number of discarded jobs is also significantly increased.

With the increase in node heterogeneity, the number of nodes with slow speed also increase. Thus, using a random policy, if a slow speed node is selected randomly, then the job is more likely to be discarded. In our algorithm , since the server is aware of the heterogeneities, it can suitably avoid a low speed node when necessary. Even in this case, there is a tendency for high-speed nodes to be overloaded and low speed nodes to be under loaded. Hence, the difference in performance.

CONCLUSION

In this paper, we have presented a distributed scheduling algorithm that can tolerate different types of system heterogeneity. Following, we have conducted several parametric studies with the objective of evaluating the effectiveness of our algorithm. Rendering its effectiveness, we have started pursuing our studies toward our goal of determining the impact of heterogeneity on the overall system performance. Our initial step has been reported here, and it concentrates on the effect of node heterogeneity. Some interesting results have been obtained. From these results, we reach the following conclusions.

- *Concerning the algorithm behavior:* the algorithm is robust to variations in the cluster size; besides, it efficiently utilizes the available state information; moreover, it is sensitive to the local scheduling policy at the processing nodes.
- *Concerning the effect of heterogeneity:* the performance of the algorithm tends to be invariant with respect to node heterogeneity; in addition, the algorithm has a large improvement over the random selection in terms of the percentage of discarded jobs.

Currently, we are studying the effects of heterogeneity in local scheduling algorithms and heterogeneities in loads on the performance of the overall system. With heterogeneities in scheduling policies, each node may autonomously decide its own scheduling policy (FCFS, SJF, etc.). Similarly, by load heterogeneities we let the external load at a node be independent of the other nodes. Similarly, each node may autonomously decide its resources and their speeds. We propose to measure the effects of such heterogeneities in terms of the response time and throughput. We conjecture that the performance of random policies will continue to deteriorate under these heterogeneities as compared to *even simple* resource allocation or execution policies.

ACKNOWLEDGEMENT

This research was sponsored in part by the NASA Langley Research Center under contracts NAG-1-1114 and NAG-1-1154.

References

[] Biyabani, S.R.; J.A. Stankovic; and K. Ramamritham. 1988. "The integration of deadline and criticalness in hard real-time scheduling." *Proc. Real-time Systems Symposium*, (DEC.), 152-160.

[] Chuang, J.Y. and J.W.S. Liu. 1988. "Algorithms for scheduling periodic jobs to minimize average error." *Proc. Real-time Systems Symposium*, (DEC.), 142-151.

[] Craig, D.W. and C.M. Woodside. 1990. "The rejection rate for tasks with random arrivals, deadlines, and preemptive scheduling." *IEEE Trans. Software Engineering SE-16,* no. 10(OCT.), 1198-1208.

[] Eager D.L.; E.D. Lazowska; and J. Zahorjan. 1986. "Adaptive load sharing in homogeneous distributed systems." *IEEE Trans. Software Engineering,* SE-12(May), no. 5, 662-675.

[] Rajkumar R.; L. Sha; and J.P. Lehoczky. 1988. "Real-time synchronization protocols for multiprocessors," *Proc. Real-time Systems Symposium.* (DEC.), 259-269.

[] Shin, K.G.; C.M. Krishna; and Y.H. Lee. "Optimal resource control in periodic real-time environments." *Proc. Real-time Systems Symposium.* (DEC.), 33-41.

[] Stankovic J. and K. Ramamritham. 1986. "Evaluation of a bidding algorithm for hard real-time distributed systems." *IEEE Trans. ComputersC-34*, no. 12(Dec.), 1130-1143.

[] ZeinElDine, O.; M. El-Toweissy; and R. Mukkamala. 1991. " A Distributed Scheduling Algorithm for Heterogeneous Real-time Systems." To appear in *Lecturer Notes in Computer Science*, Springer-Verlag.

[] Zhao, W.; K. Ramamritham; and J. Stankovic. 1987. "Scheduling tasks with resource requirements in hard-real time systems." *IEEE Trans. Software Engineering* SE-13, no. 5(May): 564-577.

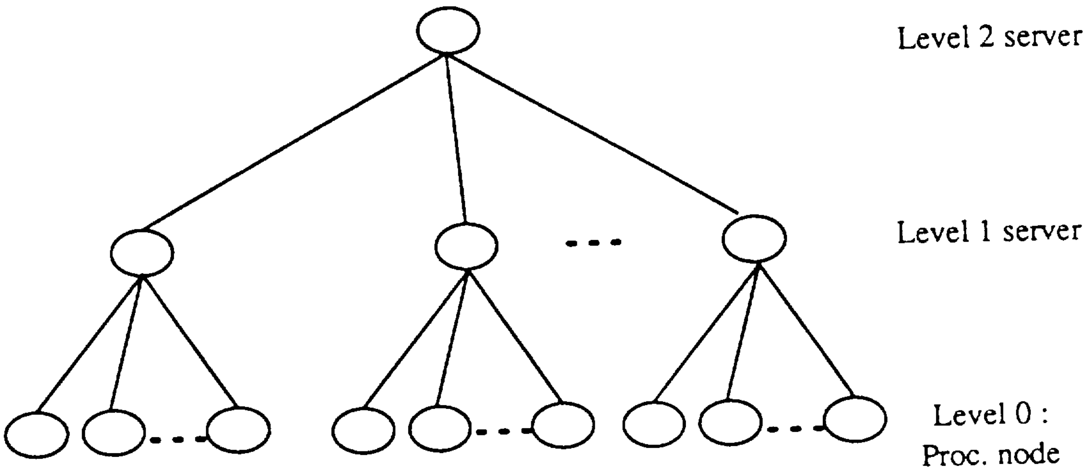

Figure 1 : System Model

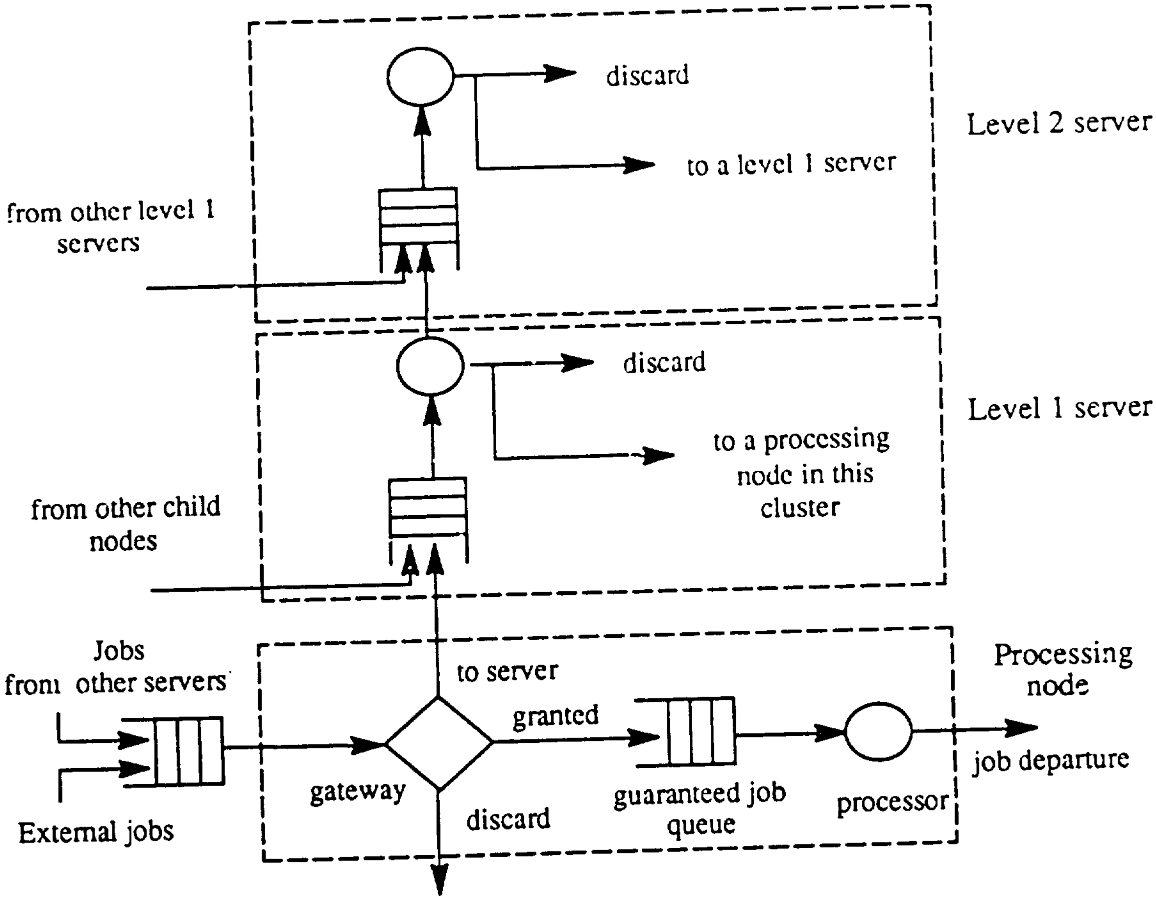

Figure 2 : Flow diagram of The Algorithm

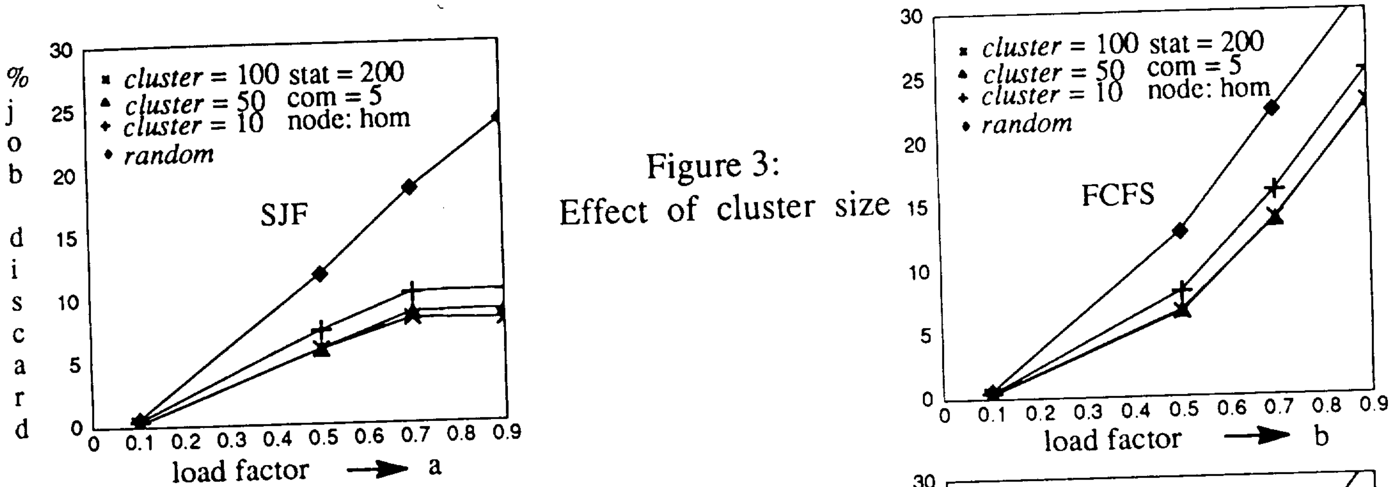

Figure 3:
Effect of cluster size

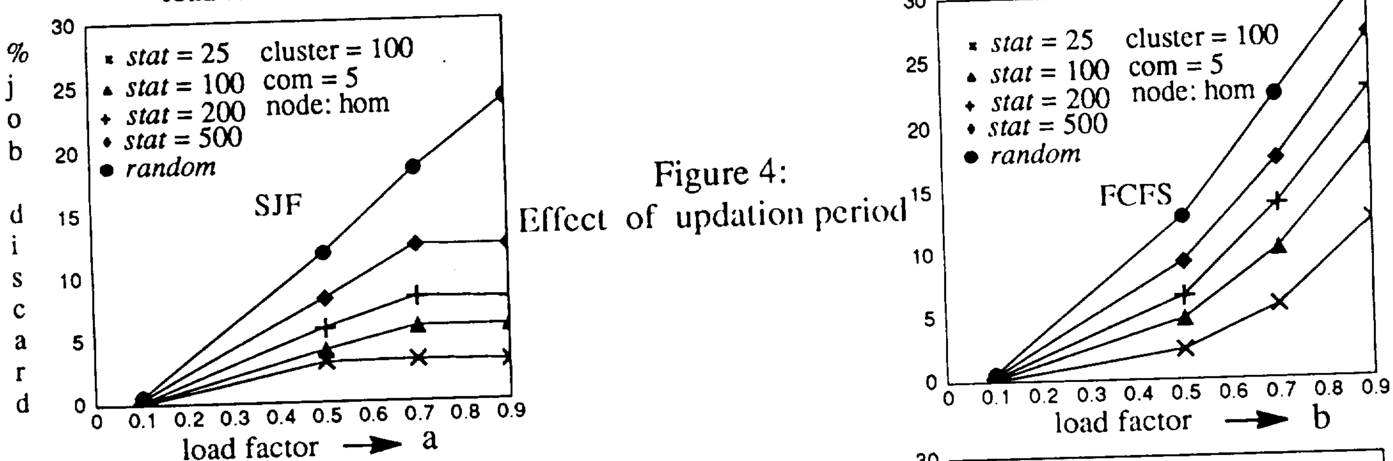

Figure 4:
Effect of updation period

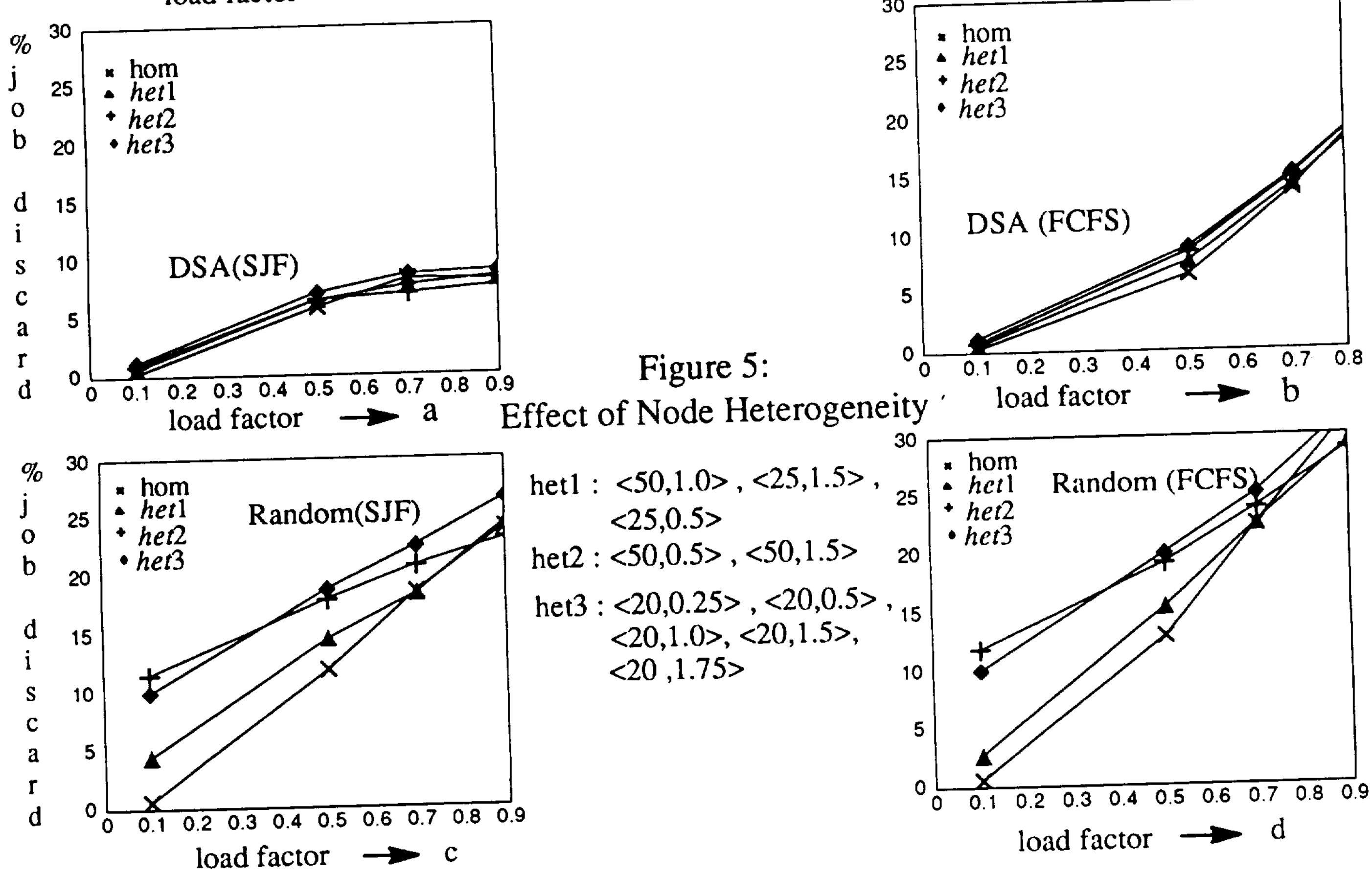

Figure 5:
Effect of Node Heterogeneity

het1 : <50,1.0> , <25,1.5> , <25,0.5>

het2 : <50,0.5> , <50,1.5>

het3 : <20,0.25> , <20,0.5> , <20,1.0>, <20,1.5>, <20 ,1.75>